processor to maintain a summary of thread activity as part of a method for providing mutual exclusion between current and next generation data elements, comprising:

determining from a first data structure on the processor's node if the processor has passed through a quiescent state;

if so, determining from a second data structure on the processor's node if all other processors on its node have passed through a quiescent state; and

if so, indicating in a third data structure accessible to all nodes that all processors on the processor's node have passed through a quiescent state.

4. (Amended) The method of claim 3 wherein if the processor determines from the first data structure on the processor's node that the processor has not passed through a quiescent state, having a callback processor check if the processor has passed through a quiescent state and, if so, having the processor indicate in the first data structure that it has passed through a quiescent state.

5. (Amended) The method of claim 3 wherein if the processor determines from the third data structure accessible to all nodes that the processor is the last processor to pass through a quiescent state, having the processor update a fourth data structure stored in the memory of each node for storing a number of the current generation of elements being processed on the node.

6. (Amended) The method of claim 3 wherein if the processor determines from the third data structure accessible to all nodes that it is the last processor to pass through a quiescent state, having a callback processor determine if there are callbacks waiting for a subsequent generation, and, if so, updating the first data structure on each node and the third data structure accessible to all nodes to indicate that all processors need to pass through an additional quiescent state.

7. (New) The method of claim 3, wherein the first, second, and third data structures are the same data structure.